The general problem
==========

The "Year 2000' or y2k problem is the failure to represent the century part of the date correctly, or at all, in computer hardware, operating systems and software packages, in date-aware embedded microcontrollers and microprocessors, and in digital data.

It has already shown up in the commercial world in credit card readers, hotel and rentacar companies checking credit or drivers' licenses by computer, and in much commercial software such as E-mail, spreadsheets and project management packages.

Computer operating systems now in use at the NRAO (some scheduled for upgrade or replacement) contain non y2k-compliant utilities. For example, although UNIX in principle has no clock problems until 2038 and a very robust calendar facility, IBM AIX 3.2.5 (which is running Charlottesville's main server right now), has a dozen non-compliant utilities associated with account management, timed shutdown, etc.

Another NRAO example is the time in the VLA on-line computer system, in which the 16-bit signed integer system date derived from the VLA clock is currently fixed algorithmically to a "cosmetic" 19YY format!! (This is not the date used to control the array, but would be the one assigned by the Modcomps to any error messages, time-stamping printouts etc. Its operational consequencies are therefore minimal, but it's a home-grown example of the sort of problem that could be widespread.)

47% of all PC's purchased from major suppliers in 1997 have firmware (RTC+BIOS) that is not fully y2k-compliant and may affect some date-aware applications even when their clocks are set properly.

Companies with "smart" office buildings are reporting failures in tests of automated thermostats, date-aware power supplies and other "smart" equipment containing date-aware microcontrollers or microprocessors. According to the US Air Force, only 10% of all date-aware microprocessors and microcontrollers sold in 1995 were y2k-compliant. Billions are sold every year!

Neither of the NRAO examples I gave above is a show-stopper. The Modcomp problem at the VLA doesn't directly affect array control, and the server in C'ville is scheduled to be replaced; its os can also be patched or upgraded even if we don't replace it. But they point to the sorts of questions that we should be asking NOW about ALL our critical systems.

We won't know for sure how big, or small, the NRAO's exposure is to y2k problems until we look.

What's "critical"?
==========

"Looking" involves taking an inventory of all our possible exposures, starting with the most critical areas first.

We need to identify the most critical areas of activity. This should be done by AD's NOW.

I suggest that we should begin by focussing on areas that are essential to keeping the telescopes operating, to keeping the essential infrastructure of the observatory functional, and to time-deadlined business (fiscal and personnel) functions.

Evaluation
==========

Then we need to take inventory of our y2k exposures, starting with these most critical areas.

The exposures may come in:

Computers -- real time clocks, firmware, operating systems, languages and compilers, applications software (in-house and commercial), databases, interfaces and device drivers.

Networks -- routers, switches, protocols, scripts and tools

Infrastructure -- all date-aware microprocessors and microcontrollers, including HVAC, office equipment, telephone systems, security systems

Telescope ops -- online computers, monitor and control software, microprocessor-controlled electronics, correlators, communications with other systems, essential materiel suppliers

Business -- payroll, personnel records, purchasing, insurance, in hardware, software and databases. Where outsourced, when will supplier be y2k compliant and what upgrades etc. will be needed for us to retain compatibility with them? Interlocking databases could be a big problem, defining when to cut over from old to new versions.

We need to evaluate the NRAO exposures in every critical date-aware area. This can be done through enquiry to suppliers or designers, by reading code, etc. and by testing.

Evaluation should eventually be by testing wherever possible. Tests may not be trivial, they will require design (both so that they are extensive enough to be valid and so that we can be reasonably sure that we can back out of them) and they will take time away from normal operations.

Systems and codes that are "y2k ready" in principle may never have had their y2k logic exercised in our operational environments, and so do have to be tested.

For complex or inaccessible systems evaluation may only be possible through testing. y2k problems in embedded processors cannot be simulated.

Testing will take planning, and take time and resources away from normal operations. We have to make this investment.

Testing can also be dangerous as well as time-consuming. Tests have to be designed so you can back out of them, or done on exact replicas of the systems.